

Oracle 数据库 10g

利用空间分析和挖掘

使应用程序更加强大

Oracle 技术白皮书

2004 年 2 月

目录

1 引言.....	1
2 Oracle 10g 中的空间分析：函数功能概述.....	2
2.1 Tiled_Bins.....	2
2.2 Aggregates_For_Geometry.....	3
2.2.1 最近邻聚集.....	3
2.2.2 距离内聚集.....	4
2.3 Aggregates_For_Layer	4
2.4 Spatial_Clusters	5
3 数据集.....	5
4 空间分析：示例应用程序.....	6
4.1 位置探测(Location Prospecting).....	6
4.2 集群分析(Clustering Analysis).....	10
4.3 基于邻近区域的估计.....	11
4.3.1 距离内聚集.....	12
4.3.2 最近邻聚集.....	13
4.3.3 加权近邻聚集.....	14
4.3.4 房地产案例.....	15
5 空间数据挖掘：框架和案例研究.....	15
5.1 分类：使用美国街区组数据的案例研究.....	16
5.2 利用空间聚集进行分类：使用美国街区组数据的案例研究.....	18
6 总结.....	20

1 引言

Oracle 有数个产品支持对仓库数据进行高效率并且有效的分析。对于探索性的分析，用户使用 OLAP/仓库技术能够通过特定维层次的上卷/下钻来分析数据。与这种 OLAP 处理不同，数据挖掘能够从数据库自动发现知识（或信息内容）。

在这样的特定环境中，Oracle 的数据挖掘 (ODM) 选项提供了几种技术，以便在 Oracle 数据库中挖掘大型数据集。这些技术包括 (1) 利用某些学习采样进行数据分类、(2) 发现数据不同属性之间隐藏的关联、(3) 识别数据中内在集群行为的集群技术。所发现的知识/行为可以有效地用于多种目的。研究性的分析工具以及数据挖掘工具在其分析中都不包括数据对象的地理位置。

为什么位置很重要？在很多应用程序中，位置可能是查明数据中隐藏模式的关键。这种情况的一个典型例子如下，1855 年的霍乱流行。当绘制出霍乱发生的位置图后，发现在这些位置的中心点附近有一个水泵；当关闭该水泵后，这种流行病就平息下去了。Tobler 的地理学第一定律恰当地表达了位置的重要性以及邻近区域的影响：

任何事物都与其他事物相关，但是邻近事物的相关性高于远的事物。

Oracle 数据库 10g 中的 Oracle Spatial 提供了新的函数功能，基于对象的位置来分析、估计和预测邻近区域的影响。可以使用 Oracle 应用服务器中的 MapViewer 对这种临近区域信息进行可视化，也可将这种信息传送到其他分析工具中。此外，临近区域信息可以扩增在数据挖掘中所使用的现有数据。在本文中，我们描述如何充分利用 Oracle Spatial 的空间分析功能来执行分析和挖掘任务。这些任务包括：

位置探测分析：使用位置探测分析，应用程序能够探测到符合特定预定义属性的地理区域。位置探测将数据分为“区块”，并为区块内的指定属性（例如：收入、年龄、消费模式）计算聚集。您可以使用这种函数功能来识别那些满足特定商业标准的区域或区块。例如，应用程序可以识别出具有较高平均收入的地理区域，并使用它们对新的商业场所的选择进行下钻研究。

集群分析：此分析函数从数据中获取共同属性，并基于数据间的相互距离将数据分成集群。例如，应用程序能够识别出犯罪率高的区域/集群。您可以在每个集群的中心部署更多警力。

基于邻近区域的估计：此统计函数基于邻近区域的已知值来估计未知值。估计函数实施了在 Tobler 的地理学第一定律中提出的空间相关性，即邻近的位置将具有相互类似的值。应用程序可以利用这些函数来估计各种变量，如基于邻近区域值的房地产价格或犯罪率。

空间挖掘：应用程序可以使用邻近区域估计来推动 Oracle 中的数据挖掘算法，如分

类和关连规则。示例包括：根据应用程序中先期的“学习”值，将新的地区分为高犯罪率地区或低犯罪率地区。

在本白皮书中，我们描述了一些执行上述各项任务的示例应用程序。这些任务将使用新推出的空间分析函数功能。在研究应用程序之前，我们在下一节中先对这些函数作一个简短的概述。

2 Oracle 10g 中的空间分析：函数功能概述

Oracle Spatial 提供了新的函数功能，将邻近区域的影响引入到探索性的或基于挖掘的分析中。下面，我们讨论在本白皮书案例研究中使用的这样一些函数。所有函数的完整列表请参见 Oracle Spatial 文档。所有这些函数都是 Oracle Spatial 中所包含的 SDO_SAM 程序包的一部分。本节可作为技术参考。如果您对 SAM 函数的技术细节不感兴趣，可以跳到下一节。

2.1 Tiled_Bins

此函数具体形式如下：

```
SDO_SAM.TILED_BINS(Lower_bound_in_dimension_1 NUMBER,  
    Upper_bound_in_dimension_1 NUMBER,  
    Lower_bound_in_dimension_2 NUMBER,  
    Upper_bound_in_dimension_2 NUMBER,  
    Tiling_level NUMBER,  
    SRID NUMBER DEFAULT NULL)  
RETURNS Table of MDSYS.SDO_REGION
```

其中 SDO_REGION 类型具有以下结构：

```
ID NUMBER  
GEOMETRY SDO_GEOMETRY
```

该函数利用区块级和边界，将x,y两个维度中低端和高端边界所覆盖的区域分为“4的 tiling_level 次方”那样多的区块(tile)。使用 SDO_REGION 类型返回这些区块（或区域）。

SDO_REGION 数据类型包括区块 id (数值型 ID 属性) 以及 SDO_GEOMETRY 类型的 GEOMETRY 属性 (有关此类型的详细信息, 请参见 Oracle Spatial 文档)。GEOMETRY 属性指定了由对应区块所覆盖的区域。如果指定了 SRID 参数, 则它表示所返回几何形状的空间参考 (坐标) 系统。

2.2 Aggregates_For_Geometry

此函数为指定的区域计算指定的聚集: “ref_geometry”。它使用 *theme_table* (主题表) 中的信息来计算这种聚集。此函数具体形式如下。

```
AGGREGATES_FOR_GEOMETRY (Theme_table VARCHAR2,  
    Theme_geom_column VARCHAR2,  
    Aggregate_type VARCHAR2,  
    Aggregate_column VARCHAR2,  
    Ref_Geometry SDO_GEOMETRY,  
    Dist_spec VARCHAR2 default NULL)  
RETURNS NUMBER
```

前两个参数指定 *theme_table* 和 *theme_geom_column*。主题表具有所需的更细或更粗级别的 (统计学) 信息。该信息用于计算对应于 *ref_geometry* 的区域的聚集。第三个参数指定聚集类型。它可以是任何数值型聚集, 如 ‘SUM’、‘MIN’、‘MAX’ 或 ‘AVG’。第四个参数指定 *theme_table* 中将要计算聚集的列。例如, 它可以是主题表的人口列。第五个参数指定将要计算聚集的 *ref_geometry*。最后一个参数 *dist_spec* 指定 *ref_geometry* 的其他参数。*dist_spec* 函数可以是 ‘sdo_num_res=N’ 或 ‘distance= <val> unit=mile’。如果是前者, 则聚集被称为最近邻近区域聚集。如果是后者, 则聚集被称为 “距离内” 聚集。

2.2.1 最近邻聚集

在这种情况下, *aggregate_for_geometry* 函数识别出 *theme_table* 中 (即其中的 *theme_geom_column*) 最接近 “ref_geometry” 的 N 行。将此集合设为 $R = \{R_1, \dots, R_n\}$, 其中 R_1, \dots, R_n 表示 *theme_table* 的 rowid。则函数返回以下值:

$$\text{Aggr_type}_{R_1 \leq r \leq R_n} (\text{aggr_col}(r))$$

例如, 如果 *aggregate_type* 设为 ‘SUM’ 并且 *aggregate_column* 设为 ‘CRIMEINDEX’, 则函数返回所指定的 N 个邻近区域的犯罪指数值的总和。

2.2.2 距离内聚集

在这种情况下, `dist_spec` 的形式为 ' distance = <d> unit=<units> ', 如果 `dist_spec` 的值为空, 则将其看作 0 距离。聚集是利用所指定距离 ' d ' 内的邻近区域来计算的。具体地说, 它识别出 `theme_table` 中的所有行, 如果该行的 ' `theme_geom_column` ' 在 ' `ref_geometry` ' 的指定距离 `d` 之内。将这些行设为集合 $R = \{R_1, R_2, \dots R_n\}$, 其中 $R_1, \dots R_n$ 表示 `theme_table` 中的 `rowid`。则此函数返回以下值:

$$\text{Aggregate_type}_{R_1 \leq r \leq R_n} ((\text{aggregate_column}(r) * \text{weight}(r))$$

注意 `weight(r)` 所表示的是第 `r` 行的加权作用, 它基于几何形状 `ref_geometry` (根据距离 `d` 进行扩展后) 与 `theme_table` 中的有效几何形状 (`theme_geom_column`) 相交的结果。图 1 显示了一个例子。图 1 中的 `ref_geometry` 与有效几何形状 (`theme_geom_col`) 只相交了 20%。因此, 我们应该按比例对此行的聚集列进行 20% 的加权。



图 1: “*Ref_Geometry*” 与有效几何形状只相交了 20%。有效几何形状的相关“聚集列”属性乘以 0.2 (20%)。

2.3 Aggregates_For_Layer

函数 `aggregates_for_geometry` 计算一个 “`ref_geometry`” 的聚集。此函数计算所指定的 “`ref_table`” 中一系列几何形状 (而不是一个特定几何形状) 的聚集。此函数具体形式如下:

```
AGGREGATES_FOR_LAYER (Theme_table VARCHAR2,  
    Theme_geom_column VARCHAR2,  
    Aggregate_type VARCHAR2,  
    Aggregate_column VARCHAR2,  
    Ref_Table VARCHAR2,  
    Ref_Geometry SDO_GEOMETRY,  
    Dist_spec VARCHAR2)
```

RETURNS Table of MDSYS.SDO_REGAGGR

其中 `SDO_REGAGGR` 类型具有以下结构:

名称	类型
REGION_ID	VARCHAR2(24)
GEOMETRY	MDSYS.SDO_GEOMETRY
AGGREGATE_VALUE	NUMBER

该函数返回 SDO_REGAGGR 对象的一个表，其中每个对象包含了使用 ref_table 的一行中的 ref_geometry 而计算出来的聚集。SDO_REGAGGR 对象将 rowid 存储在“id”属性中，将 ref_geometry 存储在“geometry”属性中，将计算的聚集存储在“aggregate_value”属性中。

2.4 Spatial Clusters

此函数计算所指定表中一系列几何形状的集群。可以执行额外的分析，以便识别集群中心或使用 Oracle Mapviewer 进行可视化。此函数具体形式如下。

```
SPATIAL_CLUSTERS (geometry_table VARCHAR2,
                  geometry_column VARCHAR2,
                  max_clusters NUMBER)
RETURNS Table of MDSYS.SDO_REGION
```

其中 SDO_REGION 类型具有以下的结构：

名称	类型
ID	NUMBER
GEOMETRY	MDSYS.SDO_GEOMETRY

此函数基于所指定几何表的 geometry_columns 来计算集群。它将每个集群以 SDO_REGION 类型的几何形状返回。ID 值被设为从 1 到最大集群数之间的数值。函数返回这些 SDO_REGION 的一个表。

3 数据集

我们使用以下的数据集（您可以将其导入到 Oracle10g 数据库中）来说明空间分析函数功能的使用情况。

犯罪率数据集：此数据集包含美国街区组的人口统计学信息和犯罪率信息，称为 USBG 数据。具体而言，这些数据包含了旧金山郡周围 10 英里内的 1500 个街区组。您可以从 US Census (美国人口普查局) 或其他商业公司那里获得此类数据。

房产值数据集：此数据集包含了关于房产及其价值的虚构数据。

4 空间分析：示例应用程序

在本节中，我们描述一些使用空间分析函数的示例应用程序。这些程序包括位置探测、集群和邻近区域估计。

4.1 位置探测(Location Prospecting)

在本应用程序中，我们将识别出具有较高平均收入的地区。我们可以利用这种地区来识别出新的商业场所。为此，我们将使用 *sdo_sam* 程序包中的 *tiled_bins* 和 *aggregates_for_layer* 函数。注意，*USBG_DATA* 表具有一列 *AVGHHICY*（户均收入），该列详细说明每个街区组的户均收入。但是这些值所针对的街区组从覆盖面积上来说通常都很小，多数情况下直径小于 0.2 英里。SAM 函数允许您使用 *tiled_bins* 函数生成适当大小的区域，并使用 *aggregates_for_layer* 函数来估计这些区域中的收入。以下是一系列用于识别高收入区域/区块的操作。

1. 使用 *tiled_bins* 函数生成适当大小的区域：我们指定用于覆盖数据集范围的低端和高端边界。我们将分块的级别指定为 2，为指定的范围生成 4*4=16 个区块。这些区块（具有 *id* 和几何形状属性）存储在表 *USBG_BINS* 中。*Tiled_bins* 函数的前四个参数指定了两个维度上的低端和高端边界（分别是经度和纬度值）。第五个参数将分块级别指定为 2，而最后一个参数将空间参考 ID (SRID) 指定为 8307（更多信息请参见 Oracle Spatial 文档）。

```
SQL> CREATE TABLE USBG_bins AS
SELECT * FROM TABLE (
sdo_sam.tiled_bins(
-122.58,
-122.15,
37.54,
37.98,
2,
8307));
```

2. 通过使用 *aggregates_for_layer* 函数，计算这 16 个区块中每个区块的总收入。此函数具有前一节中所提到的特征。

收入是使用 *USBG_DATA* 表中的细粒度街区组数据而计算出来的。前两个参数将 *USBG_DATA* 表和几何形状列指定为主题表和列。这意味着该 *USBG_DATA* 表中的信息将用于计算聚集。其后的两个参数指定聚集类型为 *SUM*，指定聚集列为 *AVGHHICY*。最后两个参数指定聚集需要计算的表和几何形状。

在本案例中,我们计算在 USBG_BINS 表中所创建区块的聚集。以下的 SQL 显示了如何获得 USBG_BINS 表中每个区块的 tile_id 和 income_per_tile 值。

```
SQL> SELECT b.id tile_id, a.aggregate_value income_per_tile
FROM TABLE(sdo_sam.aggregates_for_layer(
'USBG_DATA', 'GEOM', 'SUM', 'AVGHHICY',
'USBG_BINS', 'GEOMETRY')) a,
USBG_BINS b
WHERE b.rowid=a.region_id;
```

```
SQL> SELECT b.id tile_id, a.aggregate_value income_per_tile
FROM TABLE(sdo_sam.aggregates_for_layer(
'USBG_DATA', 'GEOM', 'SUM', 'AVGHHICY',
'USBG_BINS', 'GEOMETRY')) a,
USBG_BINS b
WHERE b.rowid=a.region_id;
TILE_ID TILED_INCOME
```

```
-----
0 807658.441
1 6054774.45
2 4363591.11
3 4568570.62
4 4153349.47
5 18242767.5
6 15879459.7
7 658047.239
8 1898098.06
9 16337.1206
10 5581901.86
11 8068864.04
13 2484907.79
14 12586631.7
15 386153.109
```

注意,如果区块与主题表的几何形状*部分相交*,则基于相交面积与 USBG_DATA 表中有效几何面积的比率,对聚集进行*按比例加权*。下图显示了一个例子。由于区块 A 与有效几何面积只相交了 20%,则有效几何形状的相关聚集乘以 0.2。

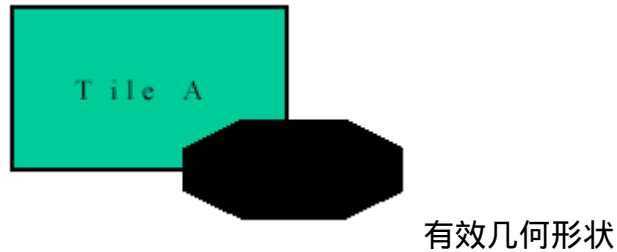


图 1 : 聚集 A 与有效几何形状只相交了 20%。有效几何形状的相关“聚集列”属性乘以 0.2 (20%)。

让我们创建一个视图 USBG_TILES，以返回以上的信息 (tile_id、收入) 以及聚集几何形状 (即区域)。

```
SQL> CREATE VIEW USBG_TILES AS
SELECT b.id tile_id, a.geometry, a.aggregate_value income_per_tile
FROM TABLE(sdo_sam.aggregates_for_layer(
'USBG_DATA', 'GEOM', 'SUM', 'AVGHHICY',
'USBG_BINS', 'GEOMETRY')) a,
USBG_BINS b
WHERE b.rowid=a.region_id;
```

3. 对区块以及三个最高收入的区块进行可视化：使用 [Oracle 应用服务器中所带的MapView](#)，我们可以基于区块的聚集收入对其进行彩色编码。图 2 显示了这些区块。底层的街区组数据以黑色表示。USBG_BINS 中的所有区块以灰色显示 (在彩色图像中为绿色)。具有最高收入的三个区块 A、B 和 C 显示带有黑色边线。可以使用以下 SQL 来获得此结果：

```
SQL> SELECT geometry
from
(select * from USBG_TILES order by tiled_income desc)
where rownum<=3;
```

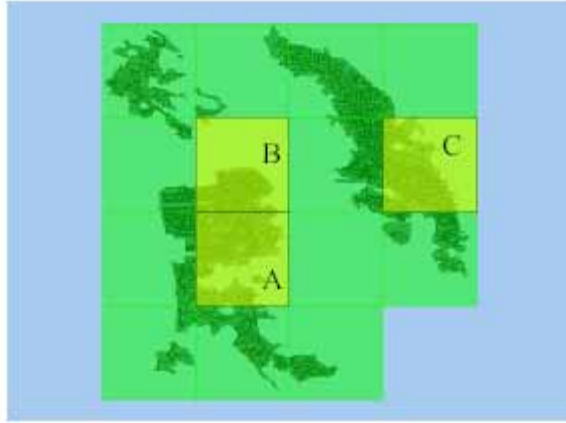


图 2：识别美国街区组数据中具有高收入的 3 个区域/区块 A、B 和 C。

4. 利用子区块分析进行下钻：通过重复步骤 1 到 3，我们可以识别出任何区块中具有高收入的子区域。例如，我们可以识别出区块 A 中具有高收入的子区域。我们在步骤 1 中指定 A 的区块边界，创建区块，并使用这些区块重复步骤 2 和 3。图 3 显示了区块 A 中具有高收入的子区块 1、2 和 3。



图 3：识别区块 A 中具有最高收入的子区域/子区块 1、2 和 3。

应用程序能够任意多次地重复此过程，以识别出适当大小的区块。

-
5. 利用 *道路* 和其他数据进行覆盖：应用程序能够利用其他令人感兴趣的信息（如道路网络）来覆盖这些区块，以便在所识别的子区块中确定最佳场所。

下面，我们将说明如何执行集群分析。

4.2 集群分析(Clustering Analysis)

应用程序可以使用 *sdo_sam* 程序包中的 *spatial_clusters* 函数，对指定表中的几何数据进行集群。此函数将表名和几何列名作为前两个参数，将集群的最大数目作为第三个参数。随后它对结果数据进行相关，以创建集群。以下一系列操作识别出高犯罪率区域的集群。

1. 将高犯罪率数据分离到一个单独的表 `USBG_HIGH_CRIMES` 中。在该表上创建一个空间索引。
- 2.

```
SQL> CREATE TABLE USBG_HIGH_CRIMES AS
SELECT * FROM USBG_DATA WHERE crimeindex > 150;
SQL> INSERT INTO user_sdo_geom._metadata
SELECT 'USBG_HIGH_CRIMES', 'GEOM', diminfo, srid
FROM user_sdo_geom._metadata
WHERE table_name='USBG_DATA';
SQL> CREATE INDEX usbg_high_crimes_sidx ON
USBG_HIGH_CRIMES(geom)
indextype is mdsys.spatial_index;
```

2. 从该 `USBG_HIGH_CRIMES` 表中获取四个集群。以下的 SQL 显示了如何获取表示集群的几何对象。注意，几何对象是包含多个数值型数组（可变量数组）属性的常规 Oracle 对象。

```
SQL > SELECT geometry
FROM table(sdo_sam.spatial_clusters(
'USBG_HIGH_CRIMES', 'GEOM', 4}}
```

3. 使用 Oracle MapViewer 对集群几何形状进行可视化。Oracle Mapviewer 是用于渲染/绘制那些存储在 Oracle Spatial 数据库中的几何对象的一个工具。图 4 以半透明的黑色显示集群。街区组显示为白色（彩色图像中为黄色），高犯罪率区域显示为黑色（彩色图像中为深蓝色）。

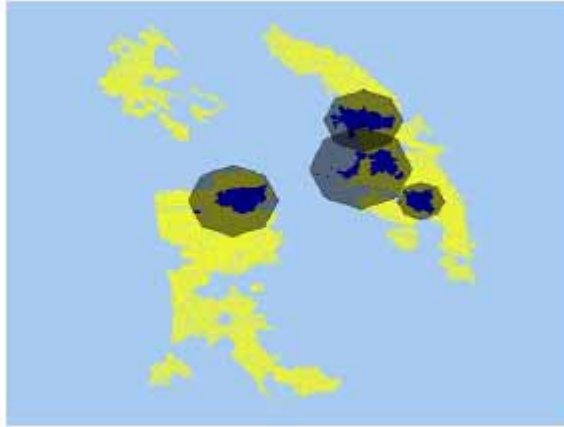


图 4：高犯罪率街区组的集群：街区组显示为白色（或黄色），高犯罪率街区组显示为蓝黑色，而集群显示为半透明的黑色。

注意，Spatial 中对几何对象的集群都是在假设相关的几何表中已有 R-树空间索引的前提下进行的。预期在以后的版本中会消除这种局限性并简化集群的使用。此外，Oracle 计划在将来的版本中支持根据非空间属性进行分组的几何集群。

我们在前面的示例中看到，Spatial 只支持在几何 (SDO_GEOMETRY) 对象上的集群。Spatial 并不对非空间维进行集群。如需要这种功能，建议用户使用 Oracle 数据挖掘产品。

4.3 基于邻近区域的估计

在本节中，我们将说明基于邻近区域值来估计属性值的函数。具体来说，我们将对美国街区组数据来进行分析。我们将该表分为两个表：通过随机采样 USBG_DATA 的 30%，我们创建 USBG_SP_TEST 表，其余部分（即 USBG_DATA 减去 USBG_SP_TEST）存储在 USBG_SP_TRAIN 表中。

我们将使用 USBG_SP_TRAIN 表中邻近区域的犯罪指数值来估计 USBG_SP_TEST 表每行的犯罪指数。我们可以使用距离内聚集或最近临近区域聚集来执行这种估计。每种方法根据距离来表示邻近区域，例如半径 2 英里，或者根据邻近区域的数目来表示，例如 5 个邻近区域。

使用 sdo_sam 程序包中的 aggregates_for_geometry 函数可以任选其中的一种方法。该函数将 USBG_SP_TEST 表中的主题表名、列名、聚集类型 (sum)、聚集列（或列字符串）和查询几何（即几何形状）作为前五个参数。

主题表是这样的表，即根据其所拥有的信息来定位邻近区域并处理其相关的聚集。在本案例中，主题表是 USBG_SP_TRAIN 表。查询几何项来自 USBG_SP_TEST 表。

4.3.1 距离内聚集

首先，让我们考虑使用 0.25 英里距离内的邻近区域来进行估计。我们将距离参数作为第 6 个参数指定给 *aggregate_for_geometry* 函数。它使用主题表的所有在指定距离内的几何项来计算聚集。我们称它为距离内聚集。

以下是 USBG_SP_TEST 的 10 行的距离内聚集。这些是使用 USBG_SP_TRAIN 表中的犯罪指数值计算出来的。注意，犯罪指数是每一千人口的犯罪率。因此，为了估计不同邻近区域的犯罪指数，我们可以求其平均值，或者可以基于人口对犯罪率进行加权。下面，我们说明如何在后一种情况中获取犯罪指数，其中所指定距离内 n 个邻近区域的犯罪指数计算如下： $\text{sum}(\text{人口} * \text{犯罪率}) / \text{sum}(\text{人口})$ 。以下的 SQL 显示了一个示例。

```
SQL> SELECT a.case_id, a.crimeindex,
(sdo_sam.aggregates_for_geometry('USBG_SP_TRAIN',
'GEOM', 'SUM', '(CRIMEINDEX*POPCY)',
a.geom, 'distance=0.25 unit=mile')/
sdo_sam.aggregates_for_geometry('USBG_SP_TRAIN',
'GEOM', 'SUM', 'POPCY',
a.geom, 'distance=0.25 unit=mile')) est_crimeindex
FROM USBG_SP_TEST a
WHERE rownum<=10;
CASE_ID CRIMEINDEX EST_CRIMEINDEX
-----
60014008002 135.91297639609 136.87794211886
60014025002 127.90569881499 128.7467151247
60750179999 87.146558483565 (null)
60750229004 104.14295487802 107.39809849166
60750203001 138.80094285833 142.61684136892
60750110002 104.62948870238 109.94998505543
60750479007 128.3307133733 128.01680754087
60014030005 95.831228330397 95.529505779919
60750215003 108.98418468505 114.92647798412
60750227005 123.75467482927 116.15322511001
```

注意，在多数情况下，所估计的犯罪指数 (*est_crimeindex*) 非常接近于实际的犯罪指数。这是因为犯罪指数在空间上是相关的（即紧邻区域的数值趋向于相互接近/并相互影响）。但是，这种方法可能会产生空值，因为在 USBG_SP_TEST 查询窗口的 0.25 英里半径内的 USBG_SP_TRAIN 中可能没有任何几何项（例如，*case_id=60750179999*）。

还要注意,如果查询点周围四分之一英里的缓冲区与主题 (USBG_SP_TRAIN) 表的几何形状只有部分相交,则基于相交面积与有效几何面积的比率,该行对聚集的作用被**成比例地加权**。这种情况如图 1 所示。

我们将估计的结果具体化为 USBG_SP_TEST 表中的 CRIME_QM 属性。此估计值是使用四分之一英里内邻近区域的犯罪指数计算出来的。我们对该估计值的均方根 (RMS) 误差的计算如下。从以下的 SQL 中,我们看到只进行空间估计可能导致犯罪指数中最多 10.7% 的误差。这意味着,只使用最近邻近区域聚集的平均预测准确率是 $100 - rms_error$, 即 89.3%。注意,467 行中只有 463 行具有非空值,也就是在四分之一英里内具有邻近区域。

```
SQL> SELECT count(*) cnt, count(crime_qm) cnt_qm,
sqrt(avg((crime_qm-crimeindex)*(crime_qm-crimeindex))) rms_error
FROM usbg_sp_test;
CNT CNT_QM RMS_ERROR
-----
467 463 10.6701902
```

4.3.2 最近邻聚集

接下来,让我们说明基于最近邻近区域的估计。将字符串 “*sdo_num_res=N*” 作为第 6 个参数指定给 *aggregates_for_geometry* 函数,为每个查询窗口的**最近 N** 个邻近区域计算聚集。我们将这种聚集称为**最近邻聚集**。下面,我们显示 USBG_SP_TEST 中 10 行的**最近邻聚集**。这些值是使用 USBG_SP_TRAIN 表中的犯罪指数值计算出来的。

```
SQL> SELECT a.case_id, a.crimeindex,
(sdo_sam.aggregates_for_geometry('USBG_SP_TRAIN',
'GEOM', 'SUM', '(CRIMEINDEX*POPCY)', a.geom,
'sdo_num_res=5')/
sdo_sam.aggregates_for_geometry('USBG_SP_TRAIN',
'GEOM', 'SUM', 'POPCY', a.geom, 'sdo_num_res=5'))
est_crimeindex
FROM USBG_SP_TEST a
```

```

WHERE rownum<=10 ;
CASE_ID CRIMEINDEX EST_CRIMEINDEX
-----
60014008002 135.91297639609 136.8652607734
60014025002 127.90569881499 129.96087246971
60750179999 87.146558483565 133.37113237008
60750229004 104.14295487802 105.8676456603
60750203001 138.80094285833 143.74645352145
60750110002 104.62948870238 107.35064837647
60750479007 128.3307133733 128.04846613008
60014030005 95.831228330397 89.435635831754
60750215003 108.98418468505 112.02556745779
60750227005 123.75467482927 117.97754465441

```

注意所估计的 *est_crimeindex* 值接近于实际的 *犯罪指数* 值。这与距离内聚集的情况相似。但是，在这种使用 *最近邻聚集* 的方法没有空值（与使用 *距离内聚集* 的方法相比较而言）。*case_id=60750179999* 的值为 133。注意，这里的差异很大，因为该例的最近邻超出了 0.25 英里。

我们将估计结果具体化为 USBG_SP_TEST 表中的 CRIME_5NBR 属性。它是使用 5 个最近邻近区域的犯罪指数计算出来的。我们对该估计值的均方根 (RMS) 误差的计算如下。从以下 SQL 中，我们看到只进行空间估计可能导致犯罪指数中最多 10% 的误差。这意味着，使用最近邻近区域聚集的平均预测准确率是 100-*rms_error*，即 90.5%。

```

SQL>SELECT count(*) cnt, count(crime_5nbr) cnt_5nbr,
sqrt(avg((crime_5nbr-crimeindex)*(crime_5nbr-crimeindex))) rms_error
FROM USBG_SP_TEST;
CNT CNT_5NBR RMS_ERROR
-----
467 467 9.49264835

```

4.3.3 加权近邻聚集

最近邻聚集的缺点是它不考虑邻近区域与查询点之间的距离，为所有邻近区域赋予相等的权重。在此函数中，我们基于邻近区域与查询点之间的距离对邻近区域的作用进行加权处理。注意，如果距离为 0，则认为它具有用户指定的“*min_distance*”值。

例如，如果有 5 个邻近区域位于距离 d_1, d_2, \dots, d_5 处，而其聚集为 a_1, a_2, \dots, a_5 ，则估计的聚集将是：

$$\text{sum}(a_1/d_1 + a_2/d_2 + \dots + a_5/d_5) / \text{sum}(1/d_1 + 1/d_2 + \dots + 1/d_5)$$

此函数仅在 10gR2 中提供。

4.3.4 房地产案例

同样，借助以上的聚集我们可以利用现有的房地产价值数据来估计新的房地产的市场价值。例如，如果现有的数据在 EXISTING_PROPERTIES 表中，而新的房地产在 NEW_PROPERTIES 表中，则我们可以使用以下两种方法来估计 NEW_PROPERTIES 中的市场价值：

_ 距离内聚集：

```
SQL> select a.property_id,
(sdo_sam.agggregates_for_geometry(
'EXISTING_PROPERTIES',
'GEOM', 'AVG', 'VALUE',
a.geom, 'distance=2 unit=mile')) est_value
from NEW_PROPERTIES a
where rownum<=10 ;
```

_ 最近邻聚集：

```
SQL> SELECT a.property_id,
(sdo_sam.agggregates_for_geometry(
'EXISTING_PROPERTIES',
'GEOM', 'AVG', 'VALUE',
a.geom, 'sdo_num_res=5')) est_value
FROM NEW_PROPERTIES a
WHERE rownum<=10 ;
```

5 空间数据挖掘：框架和案例研究

上述的 Spatial 分析函数功能可用于将邻近区域的影响具体化为应用程序数据中的附加属性。我们可以使用这种附加信息来提高数据挖掘的有效性。在利用空间影响属性（即基于邻近区域的估计）对应用程序数据进行扩增后，应用程序可以使用传统的数据挖掘工具来挖掘数据，以获得令人感兴趣的模式。图 5 显示了这种将空间分析函数与数据挖掘相结合的过程。也就是说，数据挖掘可以在扩增表数据而不是原始表数据上执行，以便产生更好的结果。

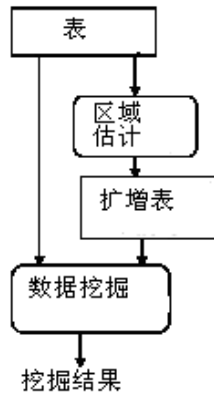


图 5：在使用空间邻近区域估计函数所获得的扩增数据上进行数据挖掘。

Oracle 的数据挖掘 (ODM) 选项提供了多种挖掘函数，以便自动检测应用程序数据中的模式。这些函数包括分类、关联规则挖掘和集群。在本节中，我们将说明一个将 ODM 的分类算法与空间分析函数相结合的案例研究。首先，我们将说明如何使用 USBG_DATA 表中的数据来执行分类。其次，我们将说明，利用附加的空间邻近区域估计对此表进行扩增（如前面的章节所述）会如何有助于改善数据挖掘的结果。在这个比较过程中，我们使用 ODM 中默认的 Naïve Bayes 分类算法。利用其他分类算法，如 Adaptive Bayes 网络，得到的比较结果基本一致。

5.1 分类：使用美国街区组数据的案例研究

为了对美国街区组数据进行分类，我们将 USBG_DATA 分为两个子集：USBG_TRAIN 和 USBG_TEST。USBG_DM_TEST 表是通过随机采样 USBG_DATA 的 30% 而形成的。USBG_TRAIN 表是使用 USBG_DATA 的其余 70% 而形成的。利用 USBG_TRAIN 数据，我们创建一个预测犯罪指数的分类模型。利用 USBG_TEST 数据，我们测试该预测的准确率。下面，我们列出在创建分类模型并测试和衡量其准确性之前的数据准备步骤。

1. *分箱 (Binning)* 详细信息请参见 ODM 文档)：对所有的列分箱，除了 USBG_TRAIN 表和 USBG_TEST 表的“case_id”和“geom”列。我们将所有属性分入 4 个“箱子”，包括目标 CRIMEINDEX 属性。
2. 删除这些表中除 case_id 列之外的所有未分箱列。现在这些表拥有 case_id 列以及其他属性的分箱列，如亚洲人、西班牙人、白人、黑人的人口和收入。
3. *创建分类模型*：
我们使用 USBG_TRAIN 表创建分类模型。为此，我们使用函数 dbms_data_mining.create_model。

第一个参数指定模型名称。第二个参数指定挖掘任务 — 是分类还是相关规则等等。第三个参数指定了用作挖掘任务学习数据集的表。第四个参数指定该表的关键字，而第五个参数指定挖掘目标属性。以下是使用 USBG_TRAIN 表的一个示例。

```
SQL>
DECLARE
BEGIN
dbms_data_mining.create_model('USBG_CLF',
'CLASSIFICATION',
'USBG_TRAIN',
'CASE_ID',
'CRIMEINDEX_BIN');
END;
/
```

4. 将分类模型应用于测试表：

接下来，我们使用所创建的模型来预测 USBG_TEST 表中街区组的值。我们使用 dbms_data_mining 程序包中的应用函数。该函数将模型名称作为第一个参数，将测试表 USBG_TEST 作为第二个参数，将主关键字 CASE_ID（该表的属性）作为第三个参数，而将结果表 USBG_TEST_RES 作为第五个参数。

```
SQL>
DECLARE
BEGIN
dbms_data_mining.apply('USBG_CLF',
'USBG_TEST',
'CASE_ID',
'USBG_TEST_RES');
END;
/
```

5. 计算模型的准确率：

可以使用函数 compute_confusion_matrix 来测量那些为测试表 USBG_TEST 进行预测的 USBG_TEST_RES 结果的准确率。该函数将结果表作为第一个参数，将测试表作为第二个参数，将测试表的主关键字作为第三个参数，将测试表中的目标属性作为第四个参数。混淆矩阵的结果存储一个表中，此表作为第五个参数而传入。该函数返回预测的准确率。以下 SQL 显示了一个例子。

```
SQL>
DECLARE
acc number;
BEGIN
dbms_data_mining.compute_confusion_matrix(acc,
'USBG_TEST_RES',
'USBG_TEST',
'CASE_ID',
'CRIMEINDEX_BIN',
'USBG_CM_TBL');
dbms_output.put_line('Accuracy = ' || TO_CHAR(acc));
END;
/
Accuracy = 0.62 (62%)
```

这表示，使用根据 USBG_TRAIN 数据所创建的模型对 USBG_TEST 表中新街区组犯罪率的预测准确率为 62%。这意味着，使用这种模型，将近 38% 的测试街区组被不正确地分类。

通过利用关于邻近区域影响的附加信息对数据进行扩增，我们可以显著提高准确率。下一小节研究了这样的一个案例。

5.2 利用空间聚集进行分类：使用美国街区组数据的案例研究

在本小节中，我们研究将“邻近区域影响”具体化为应用程序数据中的附加属性以及使用这种数据进行数据挖掘的效果。

为此，我们首先将 USBG_DATA 分为两个子集 USBG_SP_TRAIN 和 USBG_SP_TEST。USBG_SP_TEST 表是通过随机采样 USBG_DATA 的 30% 而形成的。USBG_SP_TRAIN 表是使用 USBG_DATA 的其余 70% 而形成的。我们为这两个表扩增两个属性 CRIME_5NBR 和 CRIME_QM。这些表中的 CRIME_5NBR 列显示了使用 5 个最近邻近区域所估计的犯罪率，而 CRIME_QM 列显示了使用四分之一英里聚集所估计的犯罪率。使用空间邻近区域聚集，将 USBG_SP_TRAIN 表作为主题表，从而获得估计值。

利用这两个附加属性，我们进行分类：对列分箱、创建模型、应用该模型来预测 USBG_SP_TEST 表的犯罪指数并计算准确率。

1. 分箱（详细信息请参见 ODM 文档）：分箱所有的列，除了 USBG_SP_TRAIN 和 USBG_SP_TEST 表的“case_id”和“geom.”列。我们为所有属性使用了 4 个箱子，包括目标 CRIMEINDEX 属性。对 CRIME_5NBR 和 CRIME_QM 属性也进行同样的分箱。

2. 删除这些表中除 `case_id` 列之外的所有未分箱列。现在这些表拥有 `case_id` 列以及其他属性的分箱列，如亚洲人、西班牙人、白人、黑人的人口和收入。

3. *创建分类模型：*

我们利用 `USBG_SP_TRAIN` 表创建分类模型。为此，我们使用函数 `dbms_data_mining.create_model`。第一个参数指定模型名称。第二个参数指定挖掘任务 — 是分类还是关联规则等。第三个参数指定了将要用作挖掘任务学习数据集的表。第四个参数指定该表的键，而第五个参数指定挖掘目标属性。以下是一个使用 `USBG_SP_TRAIN` 表的示例。

```
SQL> DECLARE
BEGIN
dbms_data_mining.create_model('USBG_CLF',
'CLASSIFICATION',
'USBG_SP_TRAIN',
'CASE_ID',
'CRIMEINDEX_BIN');
END;
/
```

4. *将分类模型应用于测试表：*

接下来，我们使用所创建的模型来预测 `USBG_SP_TEST` 表中街区组的值。我们使用 `dbms_data_mining` 程序包中的应用函数。该函数将模型名称作为第一个参数，将测试表 `USBG_TEST` 作为第二个参数，将主键名 `CASE_ID`（该表的属性）作为第三个参数，将结果表 `USBG_TEST_RES` 作为第五个参数。

```
SQL>
DECLARE
BEGIN
dbms_data_mining.apply('USBG_CLF',
'USBG_SP_TEST',
'CASE_ID',
'USBG_TEST_RES');
END;
/
```

5. *计算模型的准确率：*

可以使用函数 `compute_confusion_matrix` 来测量那些为测试表 `USBG_SP_TEST` 进行预测的 `USBG_TEST_RES` 结果的准确性。此函数将结果表作为第一个参数，

将测试表作为第二个参数，将测试表的主关键字作为第三个参数，将测试表中的目标属性作为第四个参数。混淆矩阵的结果存储在作为第五个参数而传入的表中。该函数返回预测的准确率。以下的 SQL 显示了一个例子。

```
SQL>
DECLARE
acc number;
BEGIN
dbms_data_mining.compute_confusion_matrix(acc,
'USBG_TEST_RES',
'USBG_SP_TEST',
'CASE_ID',
'CRIMEINDEX_BIN',
'USBG_CM_TBL');
dbms_output.put_line('Accuracy = ' || TO_CHAR(acc));
END;
/
Accuracy = 0.92 (62%)
```

这表示，使用根据 USBG_SP_TRAIN 数据所创建的模型对 USBG_SP_TEST 表中新街区组犯罪率的预测准确率为 92%。这意味着，使用这种模型，只有 8% 的测试区块被不正确地分类。

将这一准确率与前面章节中所获得的 62% 的准确率进行比较。这表明，向原始数据添加空间邻近区域信息提高了标准数据挖掘工具的预测能力。只要目标列 *在空间上相关*，这种利用空间聚集的扩增方法就特别有效。

6 总结

空间分析函数功能为在应用分析中采纳邻近区域影响提供了一种便利的机制。在使用现有数据来识别新业务的 *未来场所*、*识别数据集群*、*估计* 地产值或犯罪率时，这种函数功能特别有用。在一个特定的案例研究中，使用空间邻近区域估计而获得的估计值的准确率在 89% 左右。这种函数功能能够方便地与 Oracle 应用服务器的 MapViewer 相集成，以便对分析结果进行方便的可视化。

分析函数可以用作预处理工具，为数据挖掘任务来扩增数据。在数据挖掘任务中对空间分析函数的这种使用可以充分提高挖掘结果的准确率。在使用空间聚集的一个特定案例研究中，当挖掘数据中包含空间估计数据时，分类的准确率提高了将近 50%（从 62% 提高到 92%）。这种空间分析函数功能与 Oracle 数据挖掘功能共同作为一种功能强大的引擎，支持商务应用程序中基于位置的分析和挖掘。



版权所有 © 2004。甲骨文公司
保留所有权利

Oracle 数据库 10g
利用空间分析和挖掘使应用程序更强大
Oracle 技术白皮书
作者：Ravikanth V. Kothuri

Oracle 公司全球总部
500 Oracle Parkway Redwood Shores, CA 94065 U.S.A.
电话 650.506.7000
传真 650.506.7200
国际咨询：电话 44.932.872.020
电传 851.927444(ORACLEG)
传真 44.932.874.625
<http://www.oracle.com>